PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 11/00	A1	(11) International Publication Number: WO 99/34290
		(43) International Publication Date: 8 July 1999 (08.07.99)

(21) International Application Number: PCT/US98/26874

(22) International Filing Date: 18 December 1998 (18.12.98)

(30) Priority Data:
09/001,068 30 December 1997 (30.12.97) US

(71) Applicant: ALCATEL USA SOURCING, L.P. [US/US]; 1000 Coit Road, Plano, TX 75075 (US).

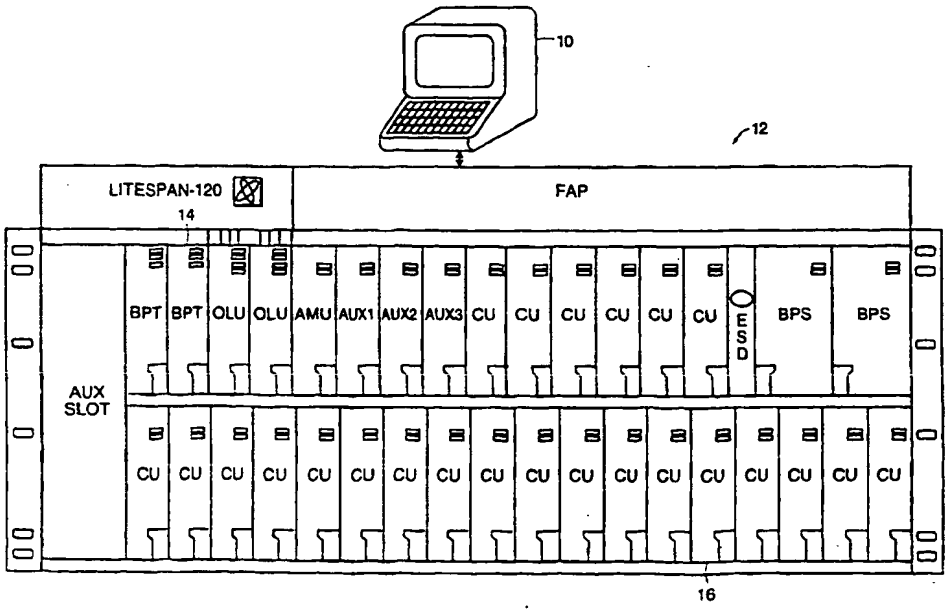
(72) Inventor: KOATH, Christopher, James; 7300 Tabor Circle, Plano, TX 75025 (US).

(74) Agent: PHILLIPS, John, C.; Fish & Richardson P.C., 601 Thirteenth Street, N.W., Washington, DC 20005 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: DEBUGGING SYSTEM



(57) Abstract

System debugging includes maintaining a list of debugging function providers in a main debug task and routing console input to a listed debugging function provider. Debugging can also include receiving information from the debugging function providers and transmitting the information to the console for display. Debugging function providers can be object-oriented debug objects. The debugging function providers can send menus of debugging functions or information produced by debugging functions. The debugging can be used to debug a telecommunications terminal.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

DEBUGGING SYSTEM

Background

This invention relates to system debugging software.

Developers write system debugging software to detect system hardware and software problems. Debugger software retrieves and displays information about the state of a system (e.g. hardware register values) and enables a developer to set the system to a specified state. Since systems include different hardware and software elements, developers often write dedicated debugger software from scratch for different systems and individual system elements.

For example, a telecommunications terminal, such as the Litespan™-2000 Access Network Unit made by DSC™ Communications Corp., includes many different hardware and software elements. A telecommunications terminal connects subscriber lines, such as POTS (Plain Old Telephone Service) and ISDN (Integrated Services Digital Network) lines, to telecommunications exchanges and other telecommunications equipment. Different hardware and software elements handle the subscriber lines, terminal processing, and transmission lines. DSC™ Communications Corp. programmed dedicated debugger software for many of these different elements.

Summary

In general, in one aspect, a method of debugging a system includes maintaining a list of debugging function providers, receiving input from a console, and routing the console input to one of the debugging function providers.

Embodiments of the method may include the following features. The method may also include receiving information from debugging function providers for console display. The debugging function providers can be

implemented in object-oriented software. The method may be used to debug a telecommunications terminal. The debugging function providers may request addition to the list of debugging function providers. The list may be displayed on a console for operator selection. The debugger function providers may supply a list of debugging functions or submenus. The debugging functions may access both hardware and software system elements.

In general, in another embodiment, a method of debugging a system includes sending information from debugging function providers to a set of main debugging routines for display on a console.

In general, in another embodiment, a system for debugging includes a console, one or more debugging function providers, and a set of main debugging routines that maintain a list of debugging function providers and carry information between the console and the debugging function providers.

Embodiments of the system for debugging may include the following features. The set of main debug routines and debugging function providers may communicate using a client/server message system. The set of main debug routines and debugging function providers may both reside on a single card such as an X2M card.

In general, in another aspect, a method of developing debug tools for debugging respective hardware and software elements under control of a user interaction with a console includes providing a common debug task that controls the console to provide a user interface that enables the user to interact with other debug objects and maintains a common programming interface for interaction with all of the debug objects, and generating debug objects for the respective system elements, each of the debug objects providing debug functions for the respective elements and functions that interact with the common debug task via the common programming interface.

Advantages may include one or more of the following.

The main debug task along with a set of console functions provide common console input/output (I/O) support for all debug objects. This reduces the amount of programming needed to create a debug object and thus reduces the development time needed to write new debugger software.

The main debug task also provides a menu of debug objects to allow a console operator to switch between different debug objects during a debugging session. This enables the console operator to see the interaction between the different system elements as they are being debugged.

Other features and advantages will become apparent from the following description, including the drawings and the claims.

Drawing Descriptions

Fig. 1 is a diagram of a console connected to hardware.

Fig. 2 is a diagram of a console, a main debug task, and a list of debug objects.

Fig. 3 is a diagram of a menu hierarchy.

Fig. 4 is a debugging session flowchart.

Detailed Description

Referring to Fig. 1, a console 10 is connected to a system 12. A developer uses the console 10 to view and change the state of the connected system 12. The system shown is a Litespan™-120 telecommunications terminal 12 made by DSC™ Communications Corp. The terminal 12 provides slots that accept different hardware cards (United States Serial No. 08/970,183 filed November 14, 1997, entitled "Telecommunications Terminal" describes a telecommunications terminal that accepts different hardware cards. The terminal 12 connects each inserted card to other inserted cards via a bus (not shown). The different cards include different components and perform different functions. For example, a Bandwidth Allocator, Processor, and Transmission (BPT) card 14 performs terminal management functions in addition to

multiplexing and demultiplexing telecommunications signals. A Channel Unit (CU) card 16 handles subscriber line signals. Many cards, including CU 16 and BPT 14, contain processors. An X2M card (a card capable of serving several E1 channels and running V5.1 protocol software) possesses strong processing capability.

Referring to Fig. 2, the debugging system includes object-oriented software instructions for console functions 19, a main debug task 18, and debug objects 20 and 22. The console functions 19 include routines that handle I/O for a variety of consoles (e.g. dedicated serial or ethernet). The main debug task 18 serves as an intermediary for passing information and commands between the console functions 10 and the debug objects 20 and 22 that provide debugging functions.

Each debug object 20 and 22 derives from a common base class (CDebug) that includes predefined characteristics. Derivation from a common base class speeds debug object 20 and 22 development. Each debug object 20 and 22 can be further programmed to perform specialized debugging functions with respect to associated hardware and software elements 24, 26. For example, one debug object 20 might access hardware registers on a POTS CU card, another debug object 22 might access memory on an ISDN CU card. Yet another debug object (not shown) might access information from software running on the system (e.g. queues, event flags, and history logs).

Though the functions provided by debug objects vary, each debug object 20 and 22 includes a PrintMenu function 26 and a HandleNewInput function 28. The PrintMenu function 26 produces menu information that allows the console operator to control debug objects 20 and 22. The PrintMenu function 26 sends this menu information to the main debug task 18 for processing with other menu information to generate menus that are displayed on the console 10.

The HandleNewInput function 28 handles console 10 input routed to the debug object 20 by the main debug task 18. For example, based on console operator input, the main debug task 18 might send to a debug object's HandleNewInput function 28 a message to execute the debug object's 20 third menu choice (not shown). The HandleNewInput function 30 calls the function corresponding to the menu choice. Each debug object 20 and 22 also includes a Constructor 24 that, upon instantiation, sends a request to the main debug task to add the debug object 20 to a main debug task list 34 of debug objects.

The main debug task 18 includes a list 34 of pointers to debug objects, a router 32, and a PrintTopMenu function 36. The router 32 sends information from the console functions 19 to debug objects 20 and 22. The router 32 also receives information produced by the debug objects 20 and 22 and sends this information to the console functions 19 for display. The router 32 and console functions 19 eliminate the need to rewrite console I/O functions or calls when developing a new debug object. The PrintTopMenu function 36 produces a menu on the console that includes a list of debug objects. The console operator can use this menu to switch among the different debug objects throughout a debugging session. This offers the console operator the convenience of debugging different system elements in a single session and the ability to debug the system as a whole as different system elements interact.

Referring to Fig. 3, the main debug task PrintTopMenu function 36 and debug object PrintMenu functions 26 and 31 allow the console operator to navigate through different menus that include lists of the different debug objects and the debugging functions they provide. The main debug task maintains a menu level variable 71 that indicates which menu should be displayed. When the menu level is zero, the main debug task displays its own menu 37, which includes a list of debug objects. When the console operator chooses one of the listed debug objects (e.g. Debug Object A) the menu level is incremented to level 1.

A debug object's PrintMenu function 26 could, for example, produce debug object menus 70 and 72. The PrintMenu function 26 could send any one of these menus to the main debug task for console display. A debug object menu 70 and 72 may include selections for debugging functions the object provides, selections for submenus, and a selection to return to a preceding menu. The PrintMenu function 26 uses the menu level variable to determine which menu to produce. For example, if the menu level is one, a debug object will produce debug object menu 70, but if the menu level is two, the debug object will produce debug object submenu 72. The PrintMenu function adjusts the menu level variable based on console operator input to let the console operator navigate through debug object menus and submenus.

Referring now to Fig. 4, at the beginning of a debugging session, the main debug task sets the menu level at zero (step 39) indicating the main debug task should display its menu of debug objects (step 40). The console operator then selects one of the debug objects (step 42) causing the main debug task to increment the menu level variable (step 44) and route further console input to the chosen debug object's HandleNewInput function.

A debug object's HandleNewInput function processes information sent by the console via the main debug task. If the console operator chooses a debugging function provided by a debug object (step 50), the HandleNewInput function calls the corresponding debug object function (52) and then reproduces the debug object menu at the current menu level for further console operator selection (step 46). If the console operator chooses a debug object submenu (step 54), the menu level variable is incremented (step 58) and the debug object PrintMenu function produces the submenu for display on the console (step 46). If the console operator chooses to return to the preceding menu, the main debug task decrements the menu level variable (step 56). If the menu level variable is zero after the decrementing (step 60), the main debug task displays its own top level menu of debug objects (step 40) and ceases forwarding console input to

the previously selected debug object. If the menu level variable is not zero after decrementing (step 60), the debug object's PrintMenu function produces the preceding debug object menu (step 46).

The different debugging objects discussed above can be distributed in many ways. For example, in a telecommunications terminal, each CU card processor could contain its own hardware debug object and communicate with a main debug task running on a BPT card. Potentially, the different processors could communicate using a client/server messaging system. Alternatively, a single card (e.g. a X2M card) could run both the main debug task and different hardware debug objects and collect hardware information from other cards over the terminal bus.

Other embodiments are within the scope of the following claims.

What is claimed is:

1. A method of debugging a system comprising:

maintaining a list of debugging function providers;

receiving input from a console connected to the system; and

routing console input to one of the listed debugging function providers for further processing.

2. The method of claim 1 further comprising

receiving information from a debugging function provider, and

transmitting the data to the console for display.

3. The method of claim 1 wherein a debugging function provider comprises an object-oriented debug object.

4. The method of claim 1 wherein further processing by debugging function providers comprises providing information for debugging a telecommunications terminal.

5. The method of claim 1 wherein further processing by debugging function providers comprises providing information for debugging a card inserted into a telecommunications terminal.

6. The method of claim 1 wherein maintaining a list of debugging function providers comprises

receiving requests from the debugging function providers for addition to the list.

7. The method of claim 1 wherein routing the console input to a debugging function provider comprises

determining which debugging function provider should receive the console input.

8. The method of claim 7 wherein determining which debugging function provider should receive the console input comprises

providing a console operator with the list of debugging function providers, and

receiving input from the console operator selecting one of the listed debugging function providers.

9. The method of claim 1 wherein further processing by a debugging function provider comprises

interacting with other software running on the system.

10. The method of claim 1 wherein further processing by a debugging function provider comprises

interacting with a hardware storage device.

11. The method of claim 9 wherein interacting with a hardware storage device comprises reading and writing hardware storage data.

12. The method of claim 1 wherein further processing by a debugging function provider comprises

producing a list of different debugging functions provided by the debugging function provider for display and console operator selection.

13. A method of debugging hardware comprising:

sending information from debugging function providers to a set of main debugging routines; and

sending the information from the set of main debugging routines to a connected console for display.

14. The method of claim 13 wherein sending information from debugging function providers comprises

sending hardware debugging information.

15. The method of claim 14 wherein sending information from debugging function providers comprises

sending software debugging information.

16. A system for debugging comprising:

a console;

one or more processors programmed to run at least one debugging function provider; and

a processor connected to the console, the processor programmed to run a set of main debug routines that accesses a list of debugging function providers and carries information between the console and the debugging function providers.

17. The system of claim 16 wherein the debugging function providers comprise object-oriented debug objects.

18. The system of claim 16 wherein the set of main debug routines comprise an object-oriented main debug task.

19. The system of claim 16 wherein the processors communicate using a client/server message system in which the set of main debug routines acts as a server and the debugging function providers as clients.

20. The system of claim 16 wherein the one or more processors that run the debugging function providers and the set of main debug routines reside in a telecommunications terminal or one or more cards inserted into the telecommunications terminal.

21. The system of claim 16 wherein a single processor includes both debugging function providers and the set of main debug routines.

22. The system of claim 21 wherein the single processor resides on an X2M card.

23. A method of debugging hardware comprising

displaying a list of debugging function providers to a console operator for selection, and

responding to console operator selection from the list.

24. The method of claim 23 wherein responding to console operator selection comprises

displaying a list of debugging functions provided by the selected debugging function provider for further console operator selection, and

responding to the further console operator selection.

25. The method of claim 24 wherein responding to the further console operator selection comprises producing a submenu for further console operator selection.

26. The method of claim 24 wherein responding to the further console operator selection comprises executing a debugging function provided by the selected debugging function provider.

27. The method of claim 26 wherein executing a debugging function comprises interacting with a telecommunications terminal system.

28. The method of claim 26 wherein executing a debugging function comprises interacting with hardware of the system.

29. The method of claim 26 wherein executing a debugging function comprises interacting with software of the system.

30. An electronic storage medium comprising instructions for causing a processor to run a set of main debug routines that maintains a list of debugging function providers and carries information between a connected console and the listed debugging function providers.

31. The electronic storage medium of 30 comprising further instructions that cause the set of main debugging routes to list debugging function providers on the console for console operator selection.

32. The electronic storage medium of 30 wherein the electronic storage medium resides on a telecommunications terminal or on a card inserted into a telecommunications terminal.

33. An electronic storage medium comprising instructions that cause a processor to run a debugging function provider that requests addition to a list of debugging function providers.

34. The electronic storage medium of claim 33 wherein the electronic storage medium resides on a telecommunications terminal or a card inserted into a telecommunications terminal.

35. A method of developing debug tools for debugging respective hardware and software elements under control of a user interacting with a console, the method comprising

providing a common debug task that controls the console to provide a user interface that enables the user interaction with respect to other debug objects and maintains a common programming interface for interaction with all of the debug objects, and

generating debug objects for the respective elements, each of the debug objects providing debug functions for the respective system elements, and functions that interact with the common debug task via the common programming interface.

36. A system of enabling a user to debug a telecommunications terminal from a console connected to the telecommunications terminal comprising

debug objects that provide debug functions for telecommunications terminal system hardware and software, and

a common debug task that controls the console to provide a user interface that enables the user interaction with respect to other debug objects and that maintains a common programming interface for interaction with all of the debug objects.

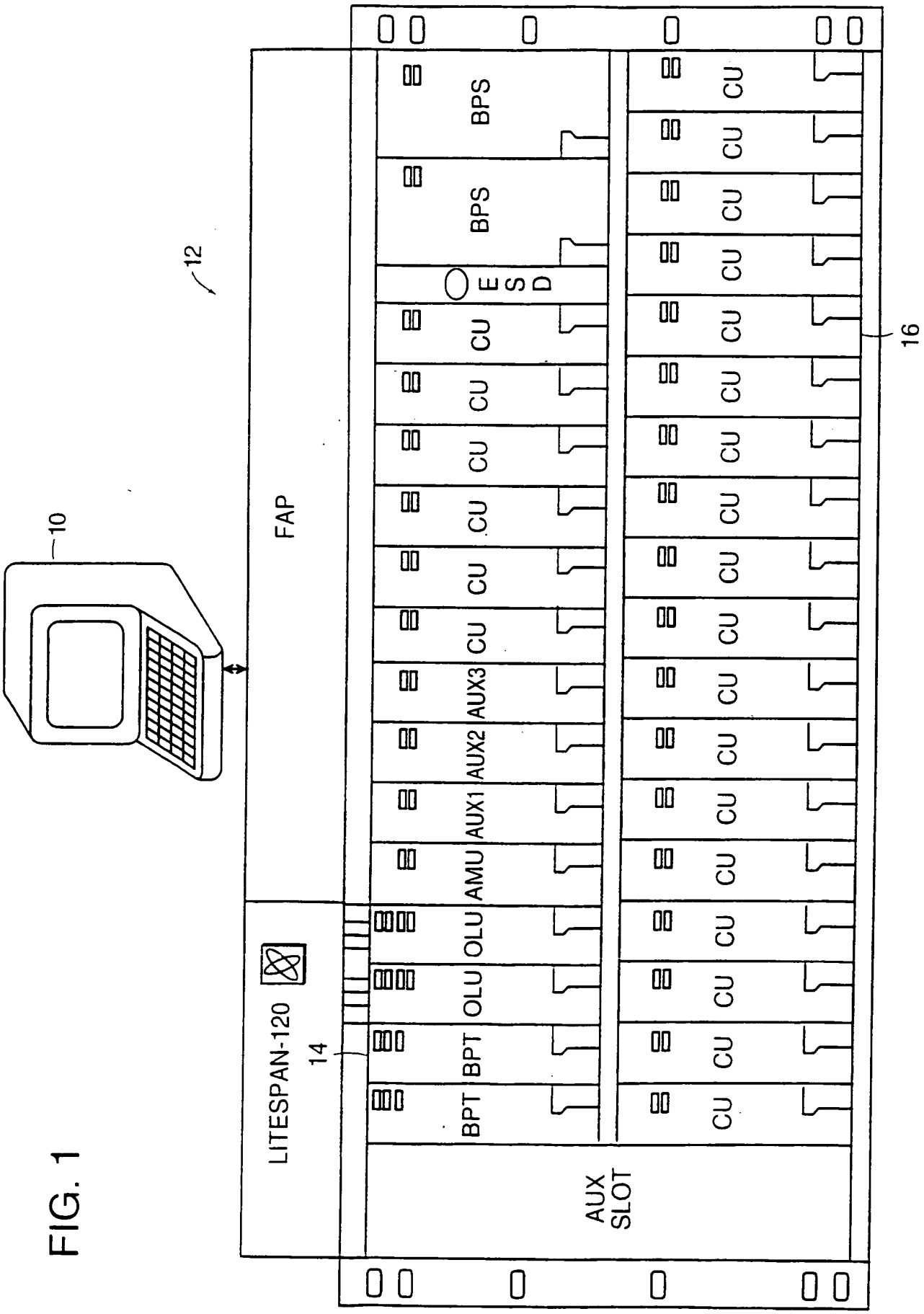

10
12
FAP
LITESPAN-120
14
16
BPS
BPS
ESD
CU
AUX3
AUX2
AUX1
AMU
OLU
OLU
BPT
BPT
AUX SLOT

FIG. 1

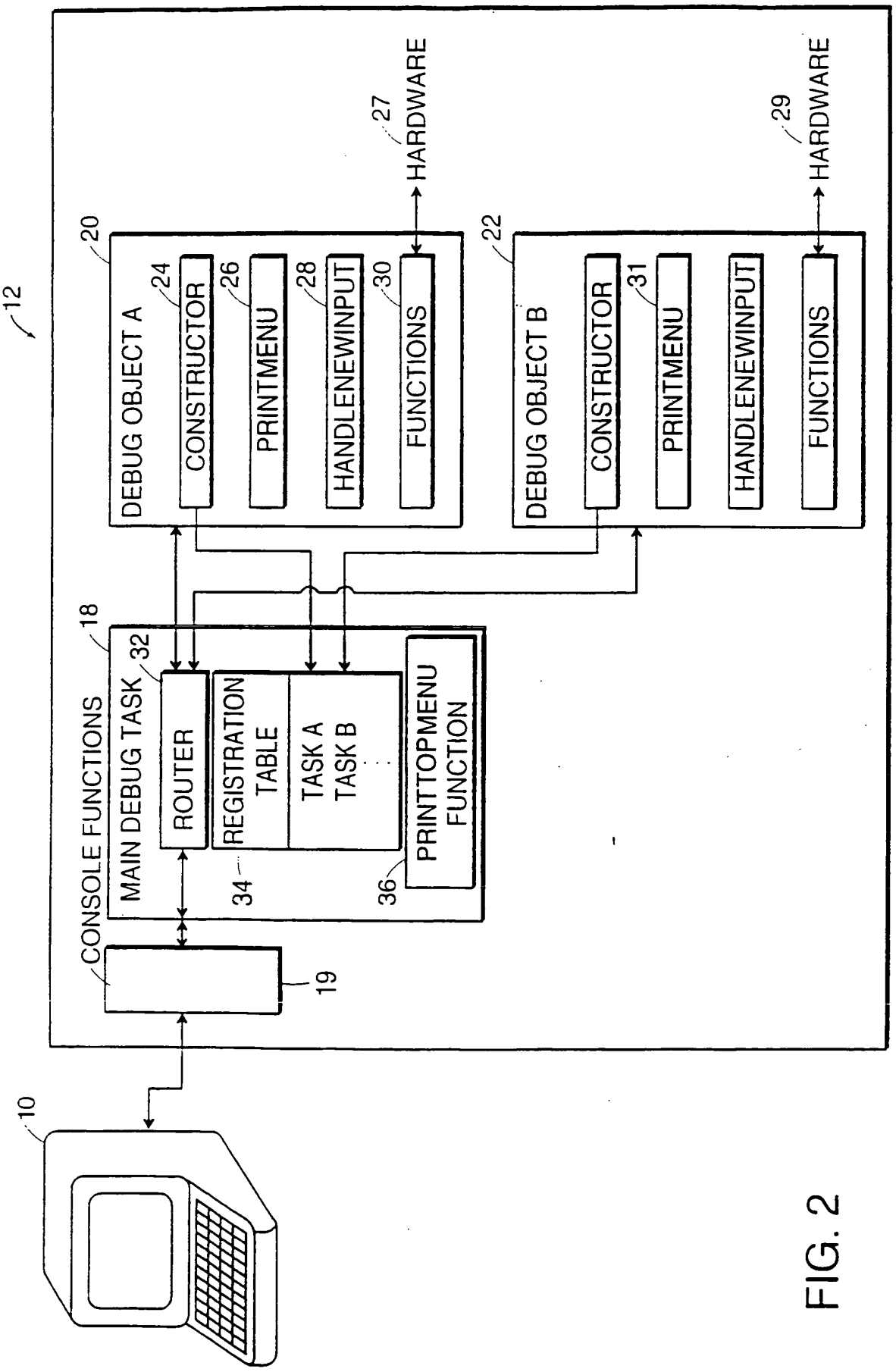

12
10
CONSOLE FUNCTIONS
19
MAIN DEBUG TASK
18
32
ROUTER
REGISTRATION TABLE
TASK A
TASK B
34
PRINTTOPMENU FUNCTION
36
DEBUG OBJECT A
20
24
CONSTRUCTOR
26
PRINTMENU
28
HANDLENEWINPUT
30
FUNCTIONS
27
HARDWARE
DEBUG OBJECT B
22
CONSTRUCTOR
31
PRINTMENU
HANDLENEWINPUT
FUNCTIONS
29
HARDWARE

FIG. 2

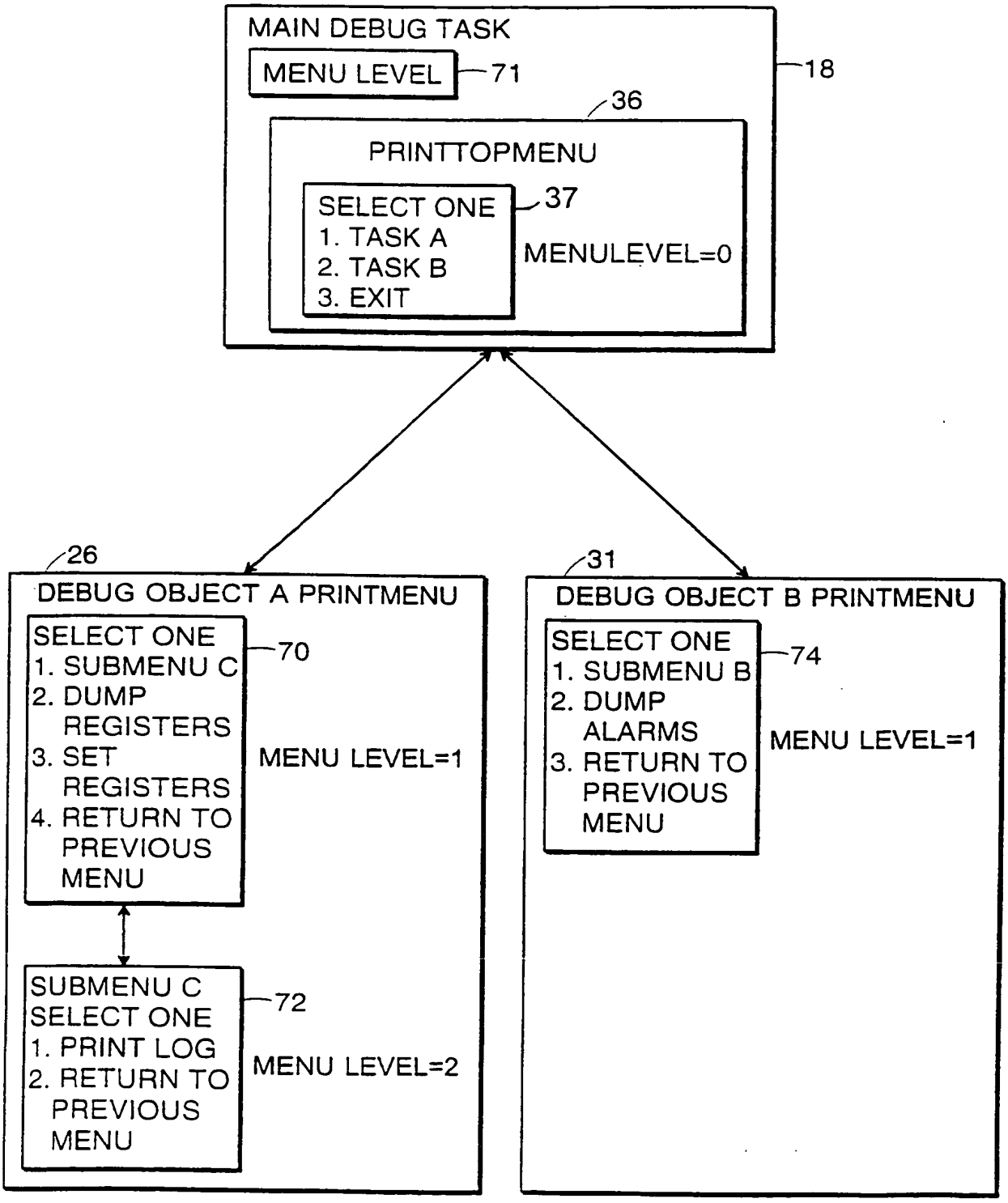

MAIN DEBUG TASK
MENU LEVEL
71
18
36
PRINTTOPMENU
SELECT ONE
1. TASK A
2. TASK B
3. EXIT
37
MENULEVEL=0
26
DEBUG OBJECT A PRINTMENU
SELECT ONE
1. SUBMENU C
2. DUMP REGISTERS
3. SET REGISTERS
4. RETURN TO PREVIOUS MENU
70
MENU LEVEL=1
SUBMENU C
SELECT ONE
1. PRINT LOG
2. RETURN TO PREVIOUS MENU
72
MENU LEVEL=2
31
DEBUG OBJECT B PRINTMENU
SELECT ONE
1. SUBMENU B
2. DUMP ALARMS
3. RETURN TO PREVIOUS MENU
74
MENU LEVEL=1

FIG. 3

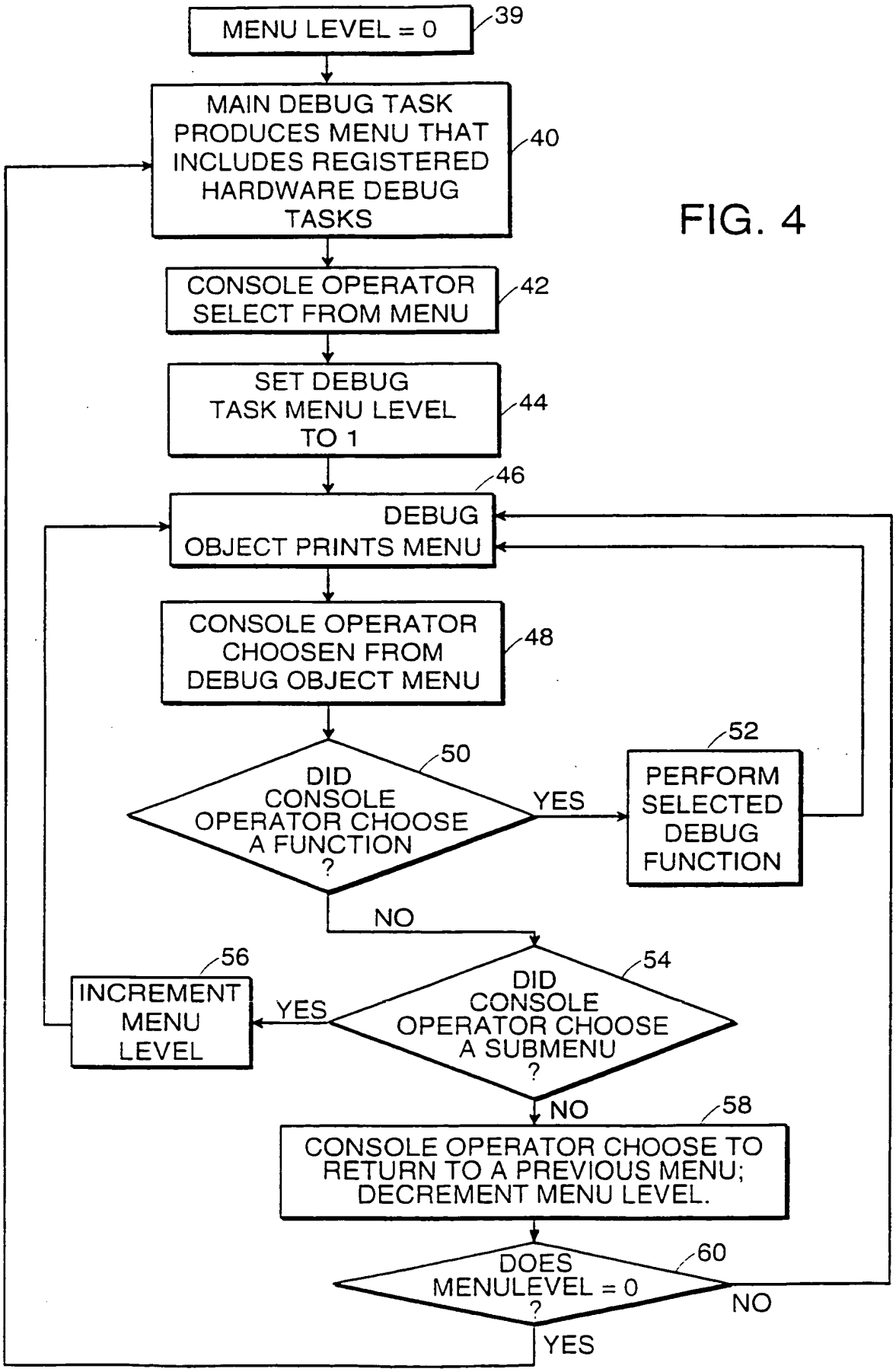

MENU LEVEL = 0
39
MAIN DEBUG TASK PRODUCES MENU THAT INCLUDES REGISTERED HARDWARE DEBUG TASKS
40
CONSOLE OPERATOR SELECT FROM MENU
42
SET DEBUG TASK MENU LEVEL TO 1
44
46
DEBUG OBJECT PRINTS MENU
CONSOLE OPERATOR CHOOSEN FROM DEBUG OBJECT MENU
48
DID CONSOLE OPERATOR CHOOSE A FUNCTION ?
50
YES
PERFORM SELECTED DEBUG FUNCTION
52
NO
INCREMENT MENU LEVEL
56
YES
DID CONSOLE OPERATOR CHOOSE A SUBMENU ?
54
NO
CONSOLE OPERATOR CHOOSE TO RETURN TO A PREVIOUS MENU; DECREMENT MENU LEVEL.
58
DOES MENULEVEL = 0 ?
60
NO
YES

FIG. 4

INTERNATIONAL SEARCH REPORT

Int.,national Application No
PCT/US 98/26874

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F11/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 814 404 A (MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.) 29 December 1997 see column 2, line 47 - column 3, line 56	1-3,7, 9-18,21, 23-31, 35,36
A	EP 0 730 227 A (SUN MICROSYSTEMS, INC.) 4 September 1996 see abstract	3,17,18

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
12 May 1999	25/05/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Corremans, G

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 98/26874

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 814404 A	29-12-1997	JP 10326203 A	08-12-1998
EP 730227 A	04-09-1996	US 5819093 A	06-10-1998
		CA 2170724 A	04-09-1996
		JP 9120366 A	06-05-1997